Workflow management in large distributed systems

2011 J. Phys.: Conf. Ser. 331 072022

(http://iopscience.iop.org/1742-6596/331/7/072022)

Journal of Physics: Conference Series **331** (2011) 072022 doi:10.1088/1742-6596/331/7/072022

Workflow management in large distributed systems

I. Legrand[1], H. Newman[1], R. Voicu[1], C. Dobre[2], C. Grigoras[3]

1) California Institute of Technology, Pasadena, CA, U.S.A.

2) Politehnica University of Bucharest, Romania

3) European Organization for Nuclear Research, Geneva, Switzerland

E-mail: *Iosif.Legrand@cern.ch*

Abstract. The MonALISA (Monitoring Agents using a Large Integrated Services Architecture) framework provides a distributed service system capable of controlling and optimizing large-scale, data-intensive applications. An essential part of managing large-scale, distributed data-processing facilities is a monitoring system for computing facilities, storage, networks, and the very large number of applications running on these systems in near real-time. All this monitoring information gathered for all the subsystems is essential for developing the required higher-level services—the components that provide decision support and some degree of automated decisions—and for maintaining and optimizing workflow in large-scale distributed systems. These management and global optimization functions are performed by higher-level agent-based services. We present several applications of MonALISA's higher-level services including optimized dynamic routing, control, data-transfer scheduling, distributed job scheduling, dynamic allocation of storage resource to running jobs and automated management of remote services among a large set of grid facilities.

1. The MonALISA architecture

The main aim for developing the MonALISA system was to provide a flexible framework capable to use in near real-time the complete monitoring information from large number of jobs, computing facilities and wide area networks to control and optimize complex workflows in distributed systems.

The MonALISA system [1] is designed as an ensemble of autonomous self-describing agent-based subsystems which are registered as dynamic services. These services are able to collaborate and cooperate in performing a wide range of distributed information-gathering and processing tasks.

An agent-based architecture [2] of this kind is well-adapted to the operation and management of large scale distributed systems, by providing global optimization services capable of orchestrating computing, storage and network resources to support complex workflows. By monitoring the state of the grid-sites and their network connections end-to-end in real time, the MonALISA services are able to rapidly detect, help diagnose and in many cases mitigate problem conditions, thereby increasing the overall reliability and manageability of the distributed computing systems. The MonALISA architecture, presented in Figure 1, is based on four layers of global services. The entire system is developed based on the Java technology.

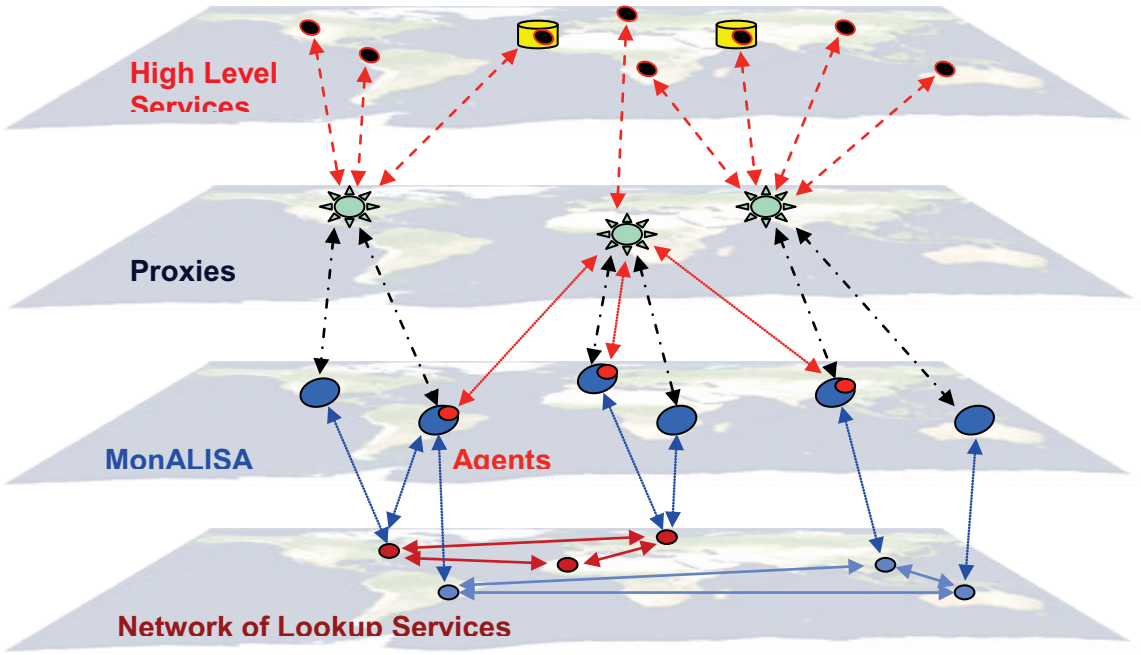

Figure 1: The four layers, main services and components of the MonALISA framework

The network of Lookup Discovery Services (LUS) provides dynamic registration and discovery for all other services and agents. MonALISA services are able to discover each other in the distributed environment and to be discovered by the interested clients. The registration uses a lease mechanism. If a service fails to renew its lease, it is removed from the LUSs and a notification is sent to all the services or other application that subscribed for such events. Remote event notification is used in this way to get a real overview of this dynamic system.

The second layer, represents the network of MonALISA service that host many monitoring tasks through the use of a multithreaded execution engine and to host a variety of loosely coupled agents that analyze the collected information in real time. The collected information can be stored locally in databases. Agents are able to process information locally and to communicate with other services or agents to perform global optimization tasks. A service in the MonALISA framework is a component that interacts autonomously with other services either through dynamic proxies or via agents that use self-describing protocols. By using the network of lookup services, a distributed services registry, and the discovery and notification mechanisms, the services are able to access each other seamlessly. The use of dynamic remote event subscription allows a service to register an interest in a selected set of event types, even in the absence of a notification provider at registration time.

The third layer of Proxy services, shown in the figure, provides an intelligent multiplexing of the information requested by the clients or other services and is used for reliable communication between agents. It can also be used as an Access Control Enforcement layer to provide secures access to the collected information.

Higher level services and client access the collected information using the proxy layer of services. A load balancing mechanism is used to allocate these services dynamically to the best proxy service. The clients, other services or agents can get any real-time or historical data by using a predicate mechanism for requesting or subscribing to selected measured values. These predicates are based on regular expressions to match the attribute description of the measured values a client is interested in. They may also be used to impose additional conditions or constraints for selecting the values. The subscription requests create a dedicated priority queue for messages. The communication with the clients is served by a pool of threads. The allocated thread performs the matching tests for all the predicates submitted by a client with the monitoring values in the data flow. The same thread is responsible to send the selected results back to the client as compressed serialized objects. Having an

independent thread for clients allows sending the information they need, in a fast and reliable way, and it is not affected by communication errors which may occur with other clients. In case of communication problems these threads will try to re-establish the connection or to clean-up the subscriptions for a client or a service which is no longer active.

Most of the monitoring tools currently used are dedicated to specific activities. Ganglia [3], and Lemon [4] are mainly used to monitor computing clusters while tools like MRTG [5], perfSONAR [6], Nagios [7] and Spectum [8] are network oriented tools providing monitor information on the network traffic, and connectivity. In the MonALISA system we provide the functionality to collect any type of information. and in this way to offer a more synergetic approach, necessary to control and optimize the execution of data intensive applications on large scale distributed systems. The framework provides integrated mechanisms to access the monitoring information from all the other MonALISA services or third party applications interfaced with the system.

Most of the monitoring tools use a set of procedures to collect the information from computing nodes or network equipment which is stored locally using XML format, the Round Robin Database (RRD) or relational databases. The stored information is than accessed using specific queries procedures by other applications to generate monitoring reports or plots. In the MonALISA system it is possible to dynamically subscribe to any class of information, to filter or generate aggregated values on the fly or to take actions based on certain values using modules that are dynamically deployed. The mechanism to select and construct the information requested by other remote services is done in memory by each MonALISA service and in this way is possible to handle significantly more information (tens of thousands) of monitoring messages per second) in near real-time.

MonALISA framework is used by several scientific communities to monitor and control large distributed systems. We present in this paper how MonALISA is used for the ALICE and CMS experiments.

2. Monitoring the CMS grid jobs

The MonALISA system is currently used in the CMS experiment to monitor all the jobs and the system where these jobs are executed. The CMS job wrappers scripts are instrumented with the ApMon library (a UDP based messaging system which is part of the MonALISA framework) that is used to send customized information for all the jobs (type of job, data set used, progress status, resources used, exit status) to several central MonALISA services. ApMon provides also the functionality to automatically report monitoring information from the systems where these jobs are executed (CPU, Load, IO traffic, diskIO network traffic). All this monitoring information from the jobs executed worldwide is collected by several MonALISA services running at CERN. The information collected by the MonALISA services is than filtered and a large set of aggregated parameters is generated on the fly by MonALISA dedicated agents. All this information is stored into a central Oracle data base running at CERN. The CMS dashboard is using this database to generate different views for the CMS job execution. All these MonALISA services and APIs run reliable without any problems for more than three years.

During the ~ last 18 months, the several CMS–dashboard MonALISA services (running at CERN) collected more 5.4 * 1010 parameters for the CMS central monitoring system (Figure 2).
Each monitored parameter for the CMS job monitoring contains 3 to 5 relative long string plus in some cases a numeric value. The average length for a “monitoring value” in CMS is ~ 240 Bytes.
The average rate of collected monitoring data for the period April-May 2010 was ~4000 parameters per second, and it collected ~2.6* 1010 monitoring values. It is for the first time we have more than 1010 monitoring values collected per one month.

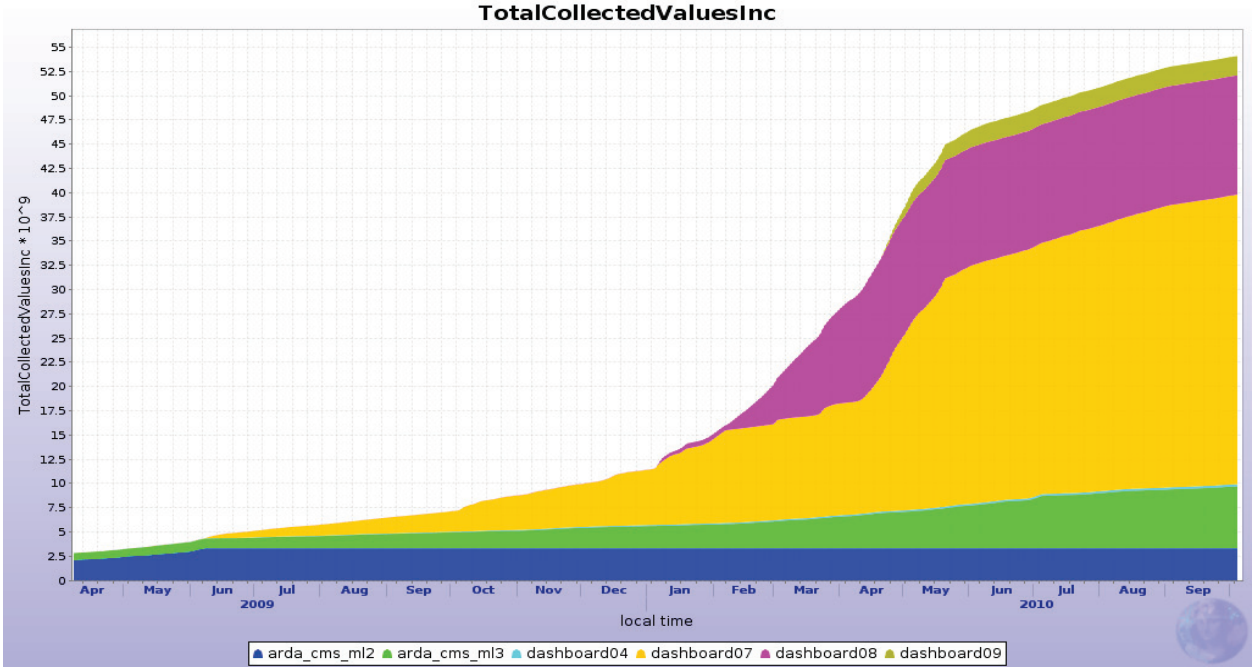

Figure 2. The total number of collected parameters by the MonALISA system for the CMS dashboard.

3. Monitoring and control for the ALICE offline activities

The ALICE computing model requires a dedicated node in each computing center that runs the management software for the local resources. The same node is also running a MonALISA service that collects monitoring information from all computing nodes, storage systems, data-transfer applications, and software running in the local cluster. This yields more than 1.1 million parameters published in MonALISA, each with an update frequency of one minute. Moreover, ALICE-specific filters aggregate the raw parameters to produce system-overview parameters in realtime. These higher-level values are usually collected, stored, and displayed in the central MonALISA Repository for ALICE and are the fuel for taking automatic actions.

The ALICE computing model matched the MonALISA architecture closely so the pieces fit naturally together, but it also provided us with a big opportunity to fulfill the project's initial goal: using the monitoring data to improve the observed system. Indeed, the actions framework implemented within MonALISA represents the first step toward the automation of decisions that can be made based on the monitoring information. It is worth noting that actions can be taken at two key points: locally, close to the data source (in the MonALISA service) where simple actions can be taken; and globally, in a MonALISA client where the logic for triggering the action can be more sophisticated, as it can depend on several flows of data. Hence, the central client is equipped with several decision-making agents that help in operating this complex system: restarting remote services when they don't pass the functional tests, sending e-mail alerts or instant messages when automatic restart procedures don't fix the problems, coordinating network-bandwidth tests between pairs of remote sites, managing the DNS-based load balancing of the central machines, and automatically executing standard applications when CPU resources are idle.

The actions framework has become a key component of the ALICE grid. In addition to monitoring the state of the various grid components and alerting the appropriate people to any problems that occur during the operation, this framework is also used to automate the processes. One such automation takes care of generating Monte Carlo data that simulates the experiment's behavior or analyzes the data. In normal cases jobs run for 10 to 12 hours and generate or analyze files on the order of 10 GB

each. ALICE jobs, however, can fail for a number of reasons: among the most frequent are network issues and local machine, storage, or central services problems. By continuously monitoring the central task queue for production jobs, the MonALISA repository takes action when the number of waiting jobs goes below the preset threshold (4,000 jobs at the moment). First, it looks to see whether any failed jobs can be rescheduled to run; then if the queue length is still too short, it will schedule new bunches of 1,000 jobs. The same framework is used to copy data automatically to remote sites and to test the network connectivity among all endpoints. Combining the continuous testing of the network and storage with error reporting has proved to be an efficient tool in debugging the system.

Having so many parameters to store and display on demand in a reasonably short time was a challenge—made even more difficult because the charts are generated on the fly based on users' options. Database response time depends on the number of values, so one step toward generating charts on demand was storing averaged values over increasing time intervals, saving space but losing resolution. Three database structures are filled in parallel: from one with high resolution that keeps data only for the last couple of months to one with very low resolution that keeps data forever. The data controller automatically selects which portions of data to extract from which structure to meet user requests, and it can fetch data from more than one structure if the request parameters demand it.

The second step to reduce the response time was spreading the queries to more than one database back end. Three identical database instances now receive all the updates while the select queries are split so that different parameters are fetched in parallel from all active back ends. Having these two options in place allows serving some 20,000 dynamic pages per day from a single front-end machine.

4. Summary

During the past seven years we have been developing a monitoring platform that provides the functionality to acquire, process, analyze, and create hierarchical structures for information on the fly in a large distributed environment. The system is based on principles that allow for scalability and reliability together with easing communication among the distributed entities. This approach to collecting any type of monitoring information in such a flexible distributed framework can be used in further developments to help operate and efficiently use distributed computing facilities.

The distributed architecture we used, without single points of failure, proved to offer a reliable distributed service system. In round-the-clock operation over the past five years we never had a breakdown of the entire system. Replicated major services in several academic centers successfully handled major network breakdowns and outages.

As of this writing, more than 350 MonALISA services are running around the clock throughout the world. These services monitor more than 20,000 computer servers, hundreds of WAN links, and tens of thousands of concurrent jobs. More than 2 million parameters are monitored in near real-time with an aggregate update rate of approximately 25,000 parameters per second. Global MonALISA repositories are used by many communities to aggregate information from many sites, properly organize them for the users, and keep long-term histories.

References

[1] MonALISA web site http://monalisa.caltech.edu
[2] MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems.
I. Legrand and all , Computer Physics Communications, Volume 180, Issue 12, p. 2472-2498.
[3] Ganglia web page: http://ganglia.info
[4] CERN cluster monitoring project LEMON http://lemon.web.cern.ch/lemon/index.shtml
[5] MRTG web page http://oss.oetiker.ch/mrtg/
[6] perfSONAR web page: http://www.perfsonar.net/
[7] Nagios web page : http://www.nagios.org/
[8] Spectrum monitoring system: http://www.ca.com/us/root-cause-analysis.aspx